SISTEMI OPERATIVI

ESERCIZIO N. 1 del 9 NOVEMBRE 2001

In una banca un **elaboratore** gestisce **C** conti correnti. L'elaboratore può essere acceduto da 2 tipi di dipendenti: i **cassieri**, in qualunque numero, e **supervisori** in numero di C dipendenti (una per conto corrente). Per motivi di sicurezza, al massimo **M** cassieri possono accedere all'elaboratore contemporaneamente. Per motivi di consistenza, quando sta accedendo un supervisore, nessun cassiere può accedere all'elaboratore, e viceversa. I supervisori invece possono accedere contemporaneamente, poiché ognuno agisce su un conto corrente diverso.
Si implementi una soluzione usando il costrutto monitor per modellare l'**elaboratore** e i processi per modellare i **cassieri** e i **supervisori** e si descriva la sincronizzazione tra i processi. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare starvation.

```
program Banca

const     M = … { numero massimo di cassiere }
type      tipo = (cassiere, supervisore);  { tipo di applicazione }

type cassiere = process
begin
      repeat
            elab.entra(cassiere);
            <usa i conti correnti>
            elab.esci(cassiere);
      until false
end

type supervisore = process(c: integer)
begin
      repeat
            elab.entra(supervisore);
            <controlla il conto c>
            elab.esci(supervisore);
      until false
end

type elaboratore = monitor

{ variabili del monitor }
var  ncassieri : integer;
      { numero di cassieri che accedono }
      nsupervisori : integer;
      { numero di supervisori che accedono }
      coda : array[tipo] of condition;
      { code su cui sospendere i dipendenti in attesa }
```

```
procedure entry entra (t: tipo)
begin
    if t = cassiere
    begin
        { se ci sono già M cassieri o c'è un supervisore }
        if ncassieri = M or nsupervisori > 0 then
            coda[t].wait;
        { occupo la risorsa }
        ncassieri++;
    end
    else { t = supervisore }
    begin
        { se ci sono dei cassieri }
        if ncassieri > 0 then
            coda[t].wait;
        { occupo la risorsa }
        nsupervisori++;
    end
end

procedure entry esci (t: tipo)
begin
    if t = cassiere
    begin
        { rilascio la risorsa }
        ncassieri--;
        { se sono l'ultimo cassiere sveglio tutti i supervisori }
        if ncassieri = 0 then
            while coda[supervisore].queue do
                coda[supervisore].signal;
        else
            { altrimenti risveglio 1 cassiere }
            coda[cassiere].signal;
    end
```

```
        else { t = supervisore }
        begin
                { rilascio la risorsa }
                nsupervisori--;
                { se sono l'ultimo supervisore sveglio M cassieri }
                if nsupervisori = 0 then
                        for i := 1 to M do
                                coda[cassiere].signal;
        end
end

begin { inizializzazione delle variabili }
        ncassieri := 0;
        nsupervisori := 0;
end

var  elab: elaboratore; { il nostro monitor }
        c1, c2, ... : cassiere;
        d1, d2, ..., dC : supervisore;

begin end.
```

Starvation

La soluzione proposta presenta starvation nel caso un tipo di dipendente non lasci mai entrare l'altro tipo.

Si può risolvere imponendo un contatore per ogni tipo di dipendente, alternando la priorità ogni tot di accessi consecutivi dello stesso tipo.